Exercise 03 Raphael Michel and Florian Stoertz

May 11, 2017

1 Linear Programs

1.1 Implementation

Using the definition given in the scipy manal, we write our graphical model as a linear program:

$$\begin{aligned} \min \quad & c^T \mu && (1)\\ \text{s.t.} \quad & A_u \mu = b_u && (2)\\ & A_e \mu = b_e && (3) \end{aligned}$$

For the given graphical model, we have

$$\begin{aligned} \mu^T = [&\mu_0(0, z_0), \mu_0(1, z_0), \mu_1(0, z_1), \mu_1(1, z_1), \mu_2(0, z_2), \mu_2(1, z_2), && (4)\\ &\mu_{01}(0, 0, z_0, z_1), \mu_{01}(0, 1, z_0, z_1), \mu_{01}(1, 0, z_0, z_1), \mu_{01}(1, 1, z_0, z_1), && (5)\\ &\mu_{02}(0, 0, z_0, z_2), \mu_{02}(0, 1, z_0, z_2), \mu_{02}(1, 0, z_0, z_2), \mu_{02}(1, 1, z_0, z_2), && (6)\\ &\mu_{12}(0, 0, z_1, z_2), \mu_{12}(0, 1, z_1, z_2), \mu_{12}(1, 0, z_1, z_2), \mu_{12}(1, 1, z_1, z_2)] && (7) \end{aligned}$$

and a cost vector of

$$\begin{aligned} c^T = [&\psi_0(0), \psi_0(1), \psi_1(0), \psi_1(1), \psi_2(0), \psi_2(1), && (8)\\ &\psi_p(0, 0), \psi_p(0, 1), \psi_p(1, 0), \psi_p(1, 1), && (9)\\ &\psi_p(0, 0), \psi_p(0, 1), \psi_p(1, 0), \psi_p(1, 1), && (10)\\ &\psi_p(0, 0), \psi_p(0, 1), \psi_p(1, 0), \psi_p(1, 1)]. && (11) \end{aligned}$$

From our constraint $\mu_{ij}(k, l) \geq 0$ we get the inequality constraint

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} \mu \geq \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}. \qquad (12)$$

For the equality constraints, we get

$$
\left[\begin{array}{c|ccc}
C & & 0 & \\
\hline
 & E & & \\
D & & E & \\
 & & & E
\end{array}\right] \mu = \left[\begin{array}{c} \mathbb{1}_{3\times 1} \\ \hline 0 \end{array}\right]. \tag{13}
$$

In this block matrix, the C part realizes the condition $\sum_k \mu_i(k) = 1$:

$$
C = \begin{bmatrix} 1 & 1 & & & & \\ & & 1 & 1 & & \\ & & & & 1 & 1 \end{bmatrix} \tag{14}
$$

And the lower part realizes the other two conditions $\sum_k \mu_{ij}(k,l) = \mu_j(l)$ and $\sum_l \mu_{ij}(k,l) = \mu_i(k)$:

$$
D = \begin{bmatrix}
-1 & & & & & \\
 & -1 & & & & \\
 & & -1 & & & \\
 & & & -1 & & \\
-1 & & & & & \\
 & -1 & & & & \\
 & & & & -1 & \\
 & & & & & -1 \\
 & & -1 & & & \\
 & & & -1 & & \\
 & & & & -1 & \\
 & & & & & -1
\end{bmatrix} \tag{15}
$$

$$
E = \begin{bmatrix} 1 & 1 & & \\ & & 1 & 1 \\ 1 & & 1 & \\ & 1 & & 1 \end{bmatrix} \tag{16}
$$

In [28]:
```
import numpy as np
from scipy.linalg import block_diag
from scipy.optimize import linprog
```

In [35]:
```
def solve(beta):
    unary00 = .1
    unary01 = .1
    unary10 = .1
    unary11 = .9
    unary20 = .9
    unary21 = .1
    c = np.array([
        unary00, unary01, unary10, unary11, unary20, unary21,
        0, beta, beta, 0,
        0, beta, beta, 0,
        0, beta, beta, 0,
```

```
    ])
    A_ub = - np.eye(18)
    b_ub = np.zeros(18)

    C = np.array([
        [1, 1, 0, 0, 0, 0],
        [0, 0, 1, 1, 0, 0],
        [0, 0, 0, 0, 1, 1]
    ])
    D = np.array([
        [-1, 0, 0, 0, 0, 0],
        [0, -1, 0, 0, 0, 0],
        [0, 0, -1, 0, 0, 0],
        [0, 0, 0, -1, 0, 0],
        [-1, 0, 0, 0, 0, 0],
        [0, -1, 0, 0, 0, 0],
        [0, 0, 0, 0, -1, 0],
        [0, 0, 0, 0, 0, -1],
        [0, 0, -1, 0, 0, 0],
        [0, 0, 0, -1, 0, 0],
        [0, 0, 0, 0, -1, 0],
        [0, 0, 0, 0, 0, -1],
    ])
    E = np.array([
        [1, 1, 0, 0],
        [0, 0, 1, 1],
        [1, 0, 1, 0],
        [0, 1, 0, 1]
    ])
    F = block_diag(E, E, E)
    A_eq = np.vstack((
        np.hstack((C, np.zeros((3, 12)))),
        np.hstack((D, F))
    ))
    b_eq = np.hstack((np.ones(3), np.zeros(12)))
    x = linprog(c, A_ub, b_ub, A_eq, b_eq)
    return x
```

In [38]:
```
# Solving for an attractive potential
solve(1.0)
```

Out[38]:
```
     fun: 1.1000000000000001
 message: 'Optimization terminated successfully.'
     nit: 17
   slack: array([ 1.,  0.,  1.,  0.,  1.,  0.,  1.,  0.,  0.,  0.,  1.,  0.,  0.,
        0.,  1.,  0.,  0.,  0.])
  status: 0
 success: True
```

```
x: array([ 1.,  0.,  1.,  0.,  1.,  0.,  1.,  0.,  0.,  0.,  1.,  0.,  0.,
        0.,  1.,  0.,  0.,  0.])
```

For the attractive potential we get a state with the minimal energy of 1.1.

```
In [37]: solve(-1.0)
```

```
Out[37]:      fun: -1.8999999999999999
          message: 'Optimization terminated successfully.'
              nit: 17
            slack: array([ 0.5,  0.5,  0.5,  0.5,  0.5,  0.5,  0. ,  0.5,  0.5,  0. ,  0. ,
                0.5,  0.5,  0. ,  0. ,  0.5,  0.5,  0. ])
           status: 0
          success: True
                x: array([ 0.5,  0.5,  0.5,  0.5,  0.5,  0.5,  0. ,  0.5,  0.5,  0. ,  0. ,
                0.5,  0.5,  0. ,  0. ,  0.5,  0.5,  0. ])
```

For the repulsive potential we get a state with a negative minimal energy. This is not a surprise, as we posed the attractiveness as a requirement for this method in the lecture.